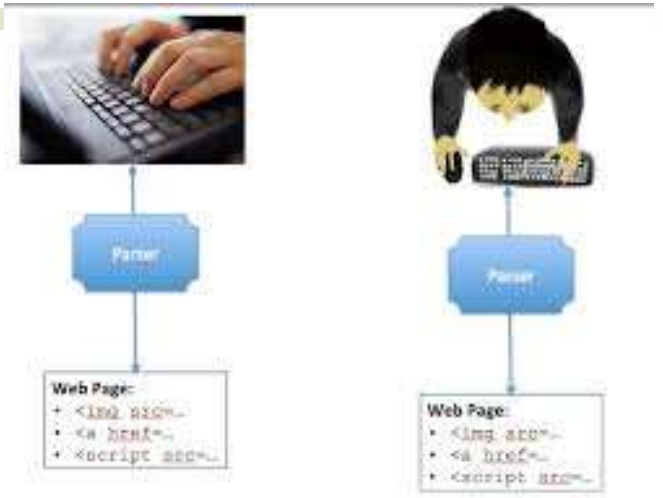
Programiranje



Nastava: prof.dr.sc. Dražena Gašpar

Datum: 18.04.2017.

Za predavanje 18.04.2017.

Za test iz teorije:

- poglavlje 5.5. (Složeni tipovi podataka)
- poglavlje 7.1. (Klasa)
- poglavlje 7.2. (Objekt)

PRVI test iz programiranja na računalu

- Imperativno programiranje

Objekti klase

Da bi se moglo raditi s objektima potrebno je

1. Napraviti (konstruirati) objekt
2. Definirati početno (inicijalno) stanje
3. Pridružiti metode objektu

JAVA rabi konstruktore za pravljenje (konstruiranje) i inicijalizaciju objekata.

Konstruktor

Pseudo metoda koja kreira objekt. To su instance metoda s UVIJEK istim imenom kao i njihove klase.

Zadaća konstruktora jeste inicijaliziranje objekta u procesu njegovog stvaranja, tako da se nakon naredbe “new” ima odmah spreman objekt za uporabu.

Metoda

Metoda je naziv dan dijelu (bloku) programskog koda koji se izvršava kao odgovor na specifičnu poruku, gdje je poruka formalna komunikacija poslana od jednog objekta drugom s ciljem izvršenja određenog servisa (usluge).

Metoda

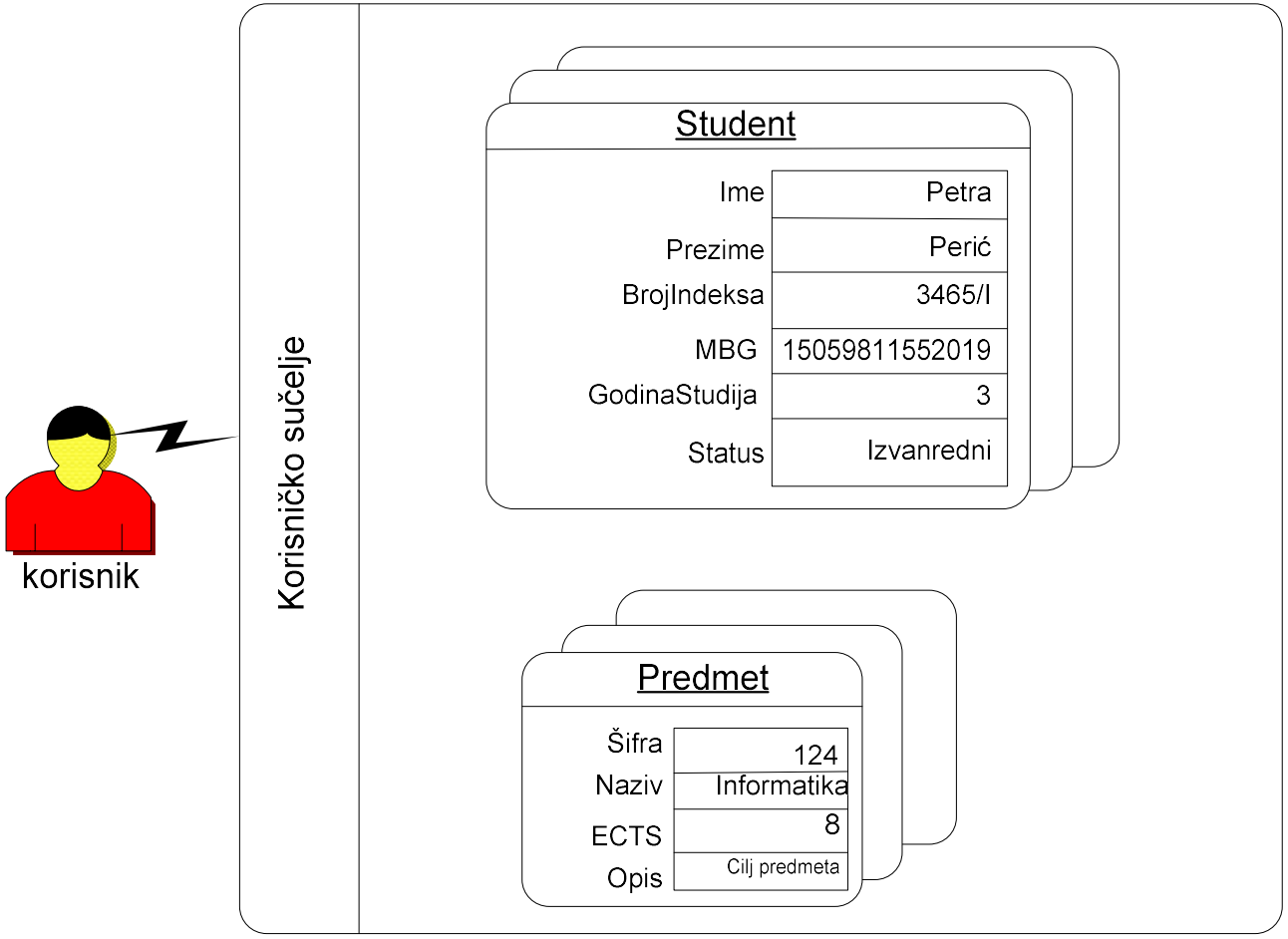
Dva su tipa poruka na koje objekt može odgovoriti:

- Zahtjev za podacima koji se naziva upit (engl.*query*)
- Zahtjev za promjenom stanja koji se naziva naredba (engl. *command*).

Skup upita i naredbi na koje će dani objekt odgovoriti naziva se sposobnost (mogućnost) objekta i određuje prigodom dizajniranja objekta, a u objektnoj paradigmi se implementira pomoću metode.

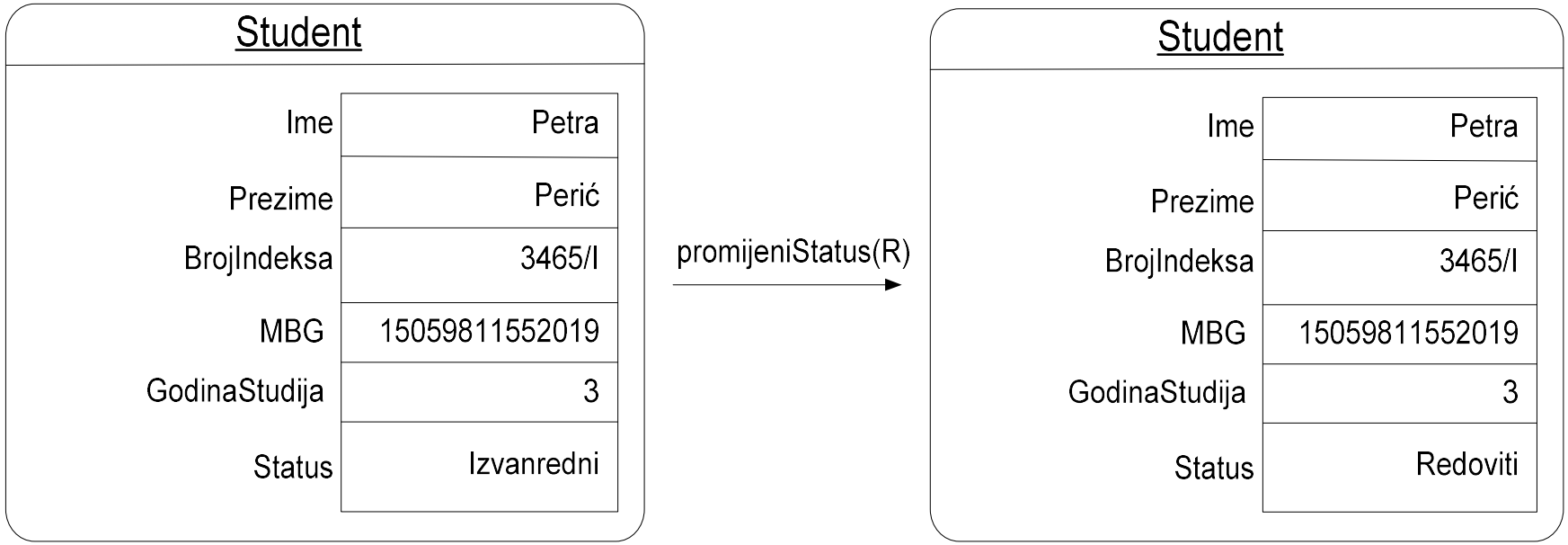
Metoda

Upit:


Korisničko sučelje
Student
Ime
Petra
Prezime
Perić
BrojIndeksa
3465/I
MBG
15059811552019
GodinaStudija
3
Status
Izvanredni
Predmet
Šifra
124
Naziv
Informatika
ECTS
8
Opis
Cilj predmeta
korisnik

Metoda

Promjena stanja objekta:



Metoda

Skup svih sposobnosti (funkcionalnosti) objekta, onako kako se vidi od strane klijenta naziva se *specifikacija*

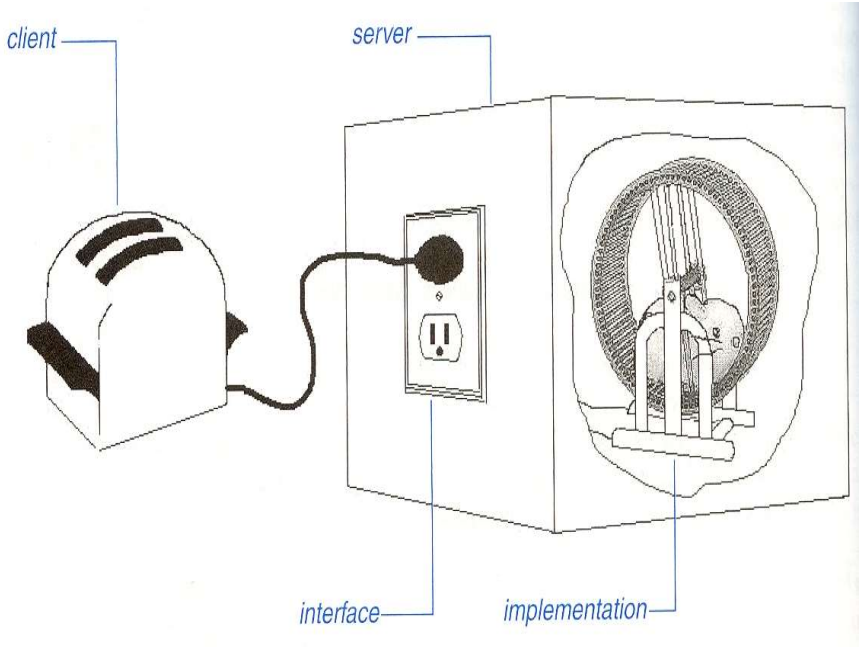
Pojam *implementacija* koristi se za opis interne strukture objekta koja ustvari čini sposobnost tj. funkcionalnost objekta

Funkcionalnost se naziva još i sučelje (engl. *interface*), ali se taj termin ne koristi da se ne bi pomiješao s Java sučeljem.

Metoda

Primjer: standardna utičnica za struju

Specifikacija jamči, ako se neki od električnih uređaja uključi u utičnicu, da će taj uređaj biti snabdjeven strujom odgovarajućeg napona. Na koji način se struja proizvodi, te kako dolazi do utičnice (implementacija) nije nešto što bi bilo koji električni uređaj trebao poznavati, ili o tome voditi računa



Metoda

Sintaksa programskog jezika Java ne dopušta odvajanje specifikacije klase od njene deklaracije.

Klasa se definira pomoću deklariranja ili definiranja klase, a sadrži i specifikacijske i implementacijske mogućnosti

```
public class Student {
    …    ←specifikacija

    public int
GodinaStudija() {
        … ←implementacija
    }
}
```

Metode

Opći oblik deklariranja metode je:

```
modifikator  tip naziv(lista parametara)
pristupa
{
        // tijelo metode
      return izraz
   }
```

Metode

Lista parametara sadrži niz parova sastavljen od tipa podatka i imena parametra, razdvojenih zarezima.

Parametri su varijable koje prihvaćaju vrijednosti argumenata proslijeđenih metodi u trenutku njenog poziva.

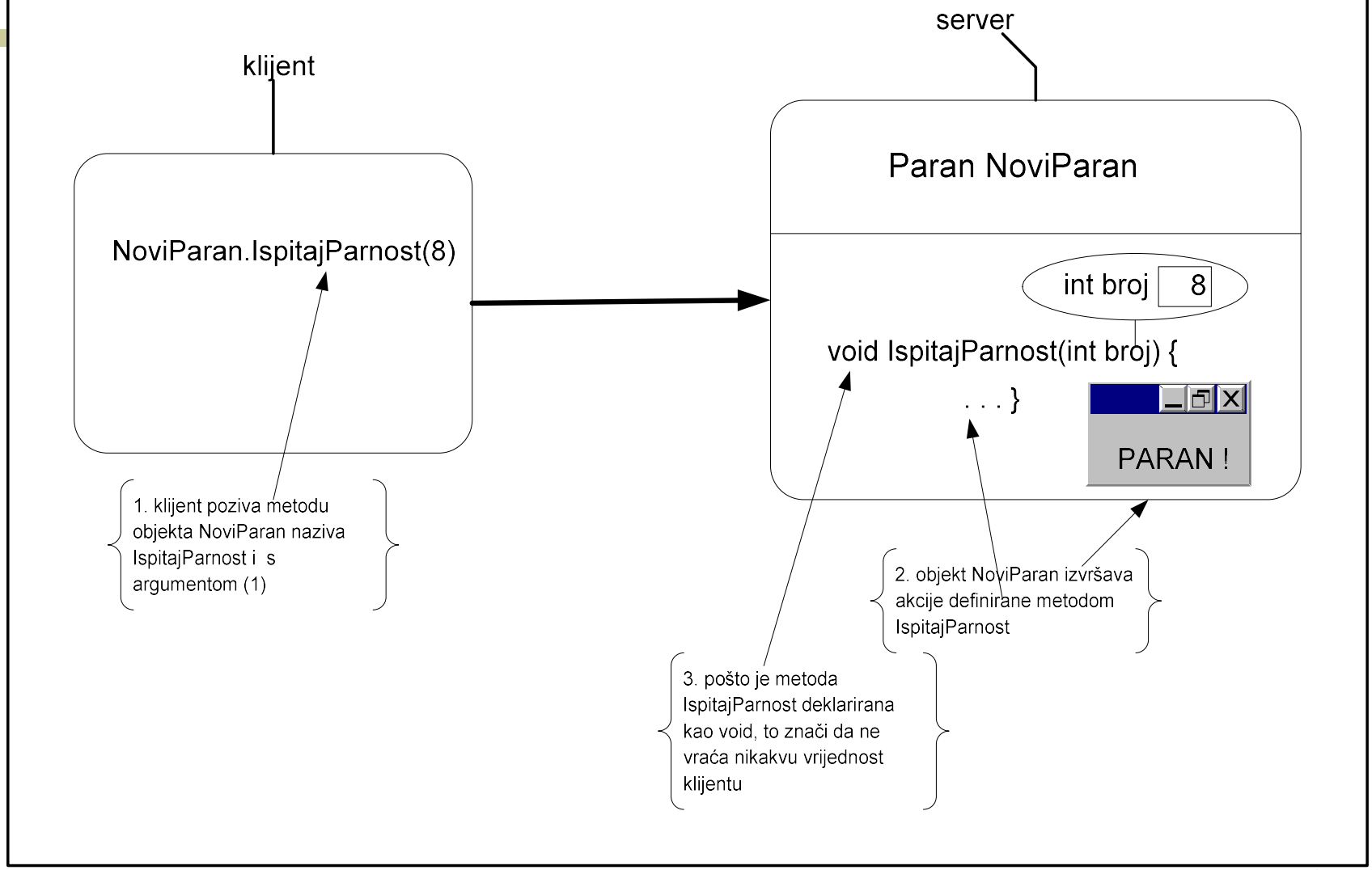
Ako metoda nema parametara, lista parametara je prazna.

Metode

Način rada objekata s metodama opisuje se općenito kao:

NazivObjekta.NazivMetode(argumenti);

Poziva se NazivMetode objekta NazivObjekta, odnosno metoda NazivMetode koja se odnosi na objekt NazivObjekta, s argumentima


server
klijent
Paran NoviParan
NoviParan.IspitajParnost(8)
int broj 8
void IspitajParnost(int broj) {
. . . }
PARAN !
1. klijent poziva metodu objekta NoviParan naziva IspitajParnost i s argumentom (1)
2. objekt NoviParan izvršava akcije definirane metodom IspitajParnost
3. pošto je metoda IspitajParnost deklarirana kao void, to znači da ne vraća nikakvu vrijednost klijentu

Metode

```
class Paran {
        void ispitajParnost (int broj)
{
         System.out.println("Broj je "+broj);
                  if (broj % 2 = = 0)
         System.out.println("PARAN !");
    }
      public static void main (String args[]) {
                  Paran NoviParan = new Paran();
                  NoviParan.ispitajParnost(1);
                  NoviParan.ispitajParnost(8);
                  NoviParan.ispitajParnost(93);
                  NoviParan.ispitajParnost(1432);
 }}
```

Zadatak: doraditi program tako da korisnik izravno unosi broj i onda se provjerava parnost. (Paran.java)

Primjer kreiranja objekta

Program: Student4.java

```
import java.util.Scanner;

public class Student4
{ public static void main(String[] args)
 { Scanner unos = new Scanner(System.in);
   System.out.print("Ime studenta ");
   String imes = unos.next();
   .....
 student student1 = new
 student(imes,prezimes,tips,indekss);
  System.out.println("Student1 "+student1.uzmiime()+"
"+student1.uzmiprezime()+" "+student1.uzmitip()+"
"+student1.uzmiindeks());  }}
```

Primjer kreiranja objekta

Program: Student4.java

```
class student
 { public student(String i, String p, String t, String n)
  { ime = i;
    prezime = p;
    tip = t;
    indeks = n;
  }
    public String uzmiime()
  { return ime; }
  public String uzmiprezime()
  { return prezime; }
  public String uzmitip()
  {return tip; }
  public String uzmiindeks()
  {return indeks; }
  private String ime, prezime, tip, indeks;   }
```

“Static” varijabla

⇒Ako se varijabla definira kao “static” to znači da postoji samo jedna takva za klasu, inače svaki objekt ima svoju kopiju svih varijabli instanci.

```
class djelatnik {
        ....
        private int id;
        private static int nextId = 1;
   }
```

“Static” varijabla

```
Public void setId() {
        id = nextId;
        nextId++;
    }

Mate.setId();
Mate.id = …;
djelatnik.nextId++;
```

“Static” varijabla

Program: Student4.java

Nadograditi s automatskim izračunom šifre za studenta i prikazom sljedeće slobodne šifre.

Rezultat: Student5.java

“Static” varijabla

Static varijable su rijetke, za razliku od static konstanti

```
public class Matematika  {
        …
 public static final double
PI=3.14159265358979;
      ….
   }

Matematika.PI   // poziv konstante
```

Kreiranje objekata

ZAPAMTITI

- **Konstruktor ima isto ime kao i klasa**
- **Klasa može imati više konstruktora**
- **Konstruktor može imati nijedan ili jedan i više parametara**
- **Konstruktor ne vraća vrijednost**
- **Konstruktor se uvijek poziva s operatorom “new”**

Kreiranje objekata

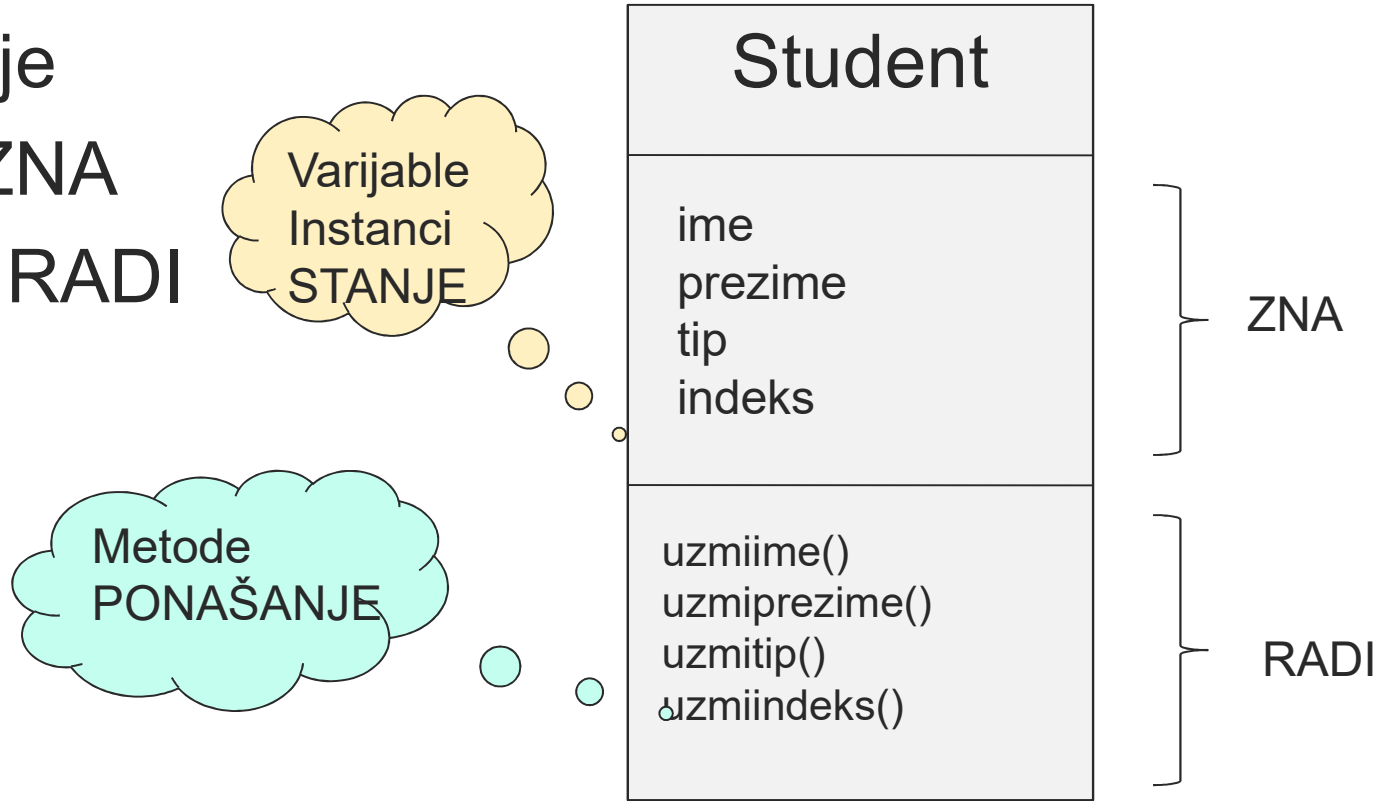
Metode

VAŽNO:

Svaka od metoda može pristupiti privatnim varijablama instanci putem njihovog imena. To je zbog toga što su varijable instanci uvijek vidljive za metode njihove vlastite klase.

Metode

Klasa opisuje
što Objekt ZNA
i što Objekt RADI



Zadatak

Formirati polje dimenzije [3] naziva Izvanredni

Popuniti s tri objekta tipa student

Klasi student dodati školarinu

Dodati metodu kojoj se parametarski šalje postotak uvećanja školarine

Program:Student6.java

Prosljeđivanje argumenata

U osnovi 2 načina:

- **po vrijednosti (call-by-value)**

  vrijednost argumenta se kopira u formalni parametar metode, promjene izvedene na parametru u metodi nemaju utjecaja na argument korišten za njegovo pozivanje

- **po referenci (call-by-reference)**

  parametru se prosljeđuje referenca na argument, a ne njegova vrijednost, a koristi se unutar metode za pristupanje stvarnom argumentu zadanom u pozivu.

Prosljeđivanje argumenata
po vrijednosti (call-by-value)

- **za primitivne tipove parametara (brojeve, Boolean i sl.)**

Metoda prima kopiju svih vrijednosti parametara i ona ne može promijeniti njihov sadržaj

Primjer:

```
public static void utrostruči (double x)
{ x = 3 * x; }
Poziv:
….
double postotak = 10;
utrostruči(postotak);
```

Prosljeđivanje argumenata
po vrijednosti (call-by-value)

Redoslijed događanja:

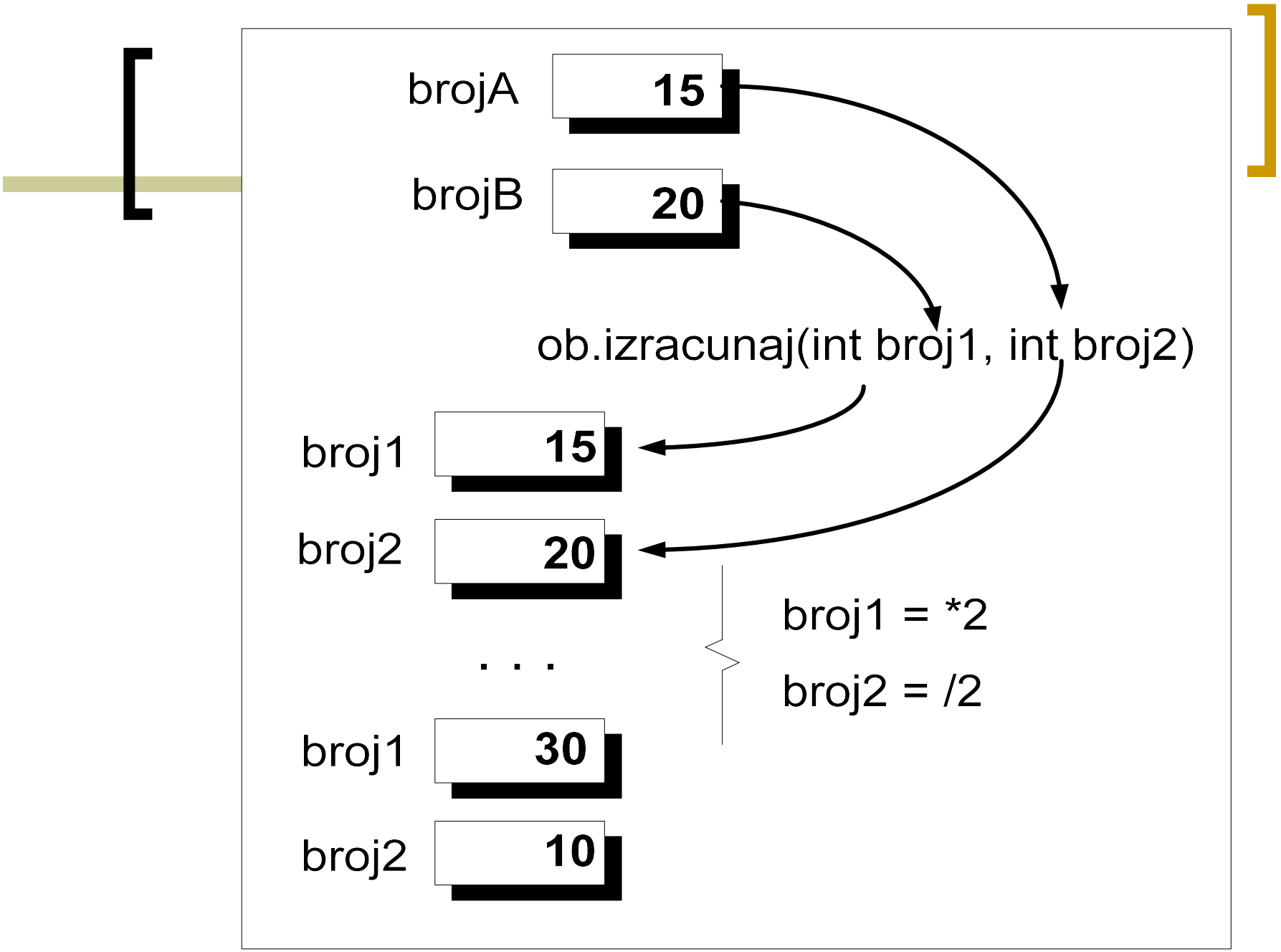
1. **“x” se inicijalizira s kopijom vrijednosti “postotak” (x= 10)**
2. **“x” se utrostruči (x=30), ali “postotak” je i dalje 10**
3. **Metoda završava i parametarska varijabla “x” više nije u uporabi.**

PozivPoVrijednosti.java – Blok za pisanje

Datoteka Uređivanje Formatiranje Prikaz Pomoć

```
// Jednostavni tipovi se prosljeđuju po vrijednosti.
import java.util.Scanner;

class Test {
  void racunaj(int a, int b) {
    a *= 2;
    b /= 2;
  System.out.println("a i b iz metode: " +
                     a + " " + b);
  }
}

class PozivPoVrijednosti {
  public static void main(String args[]) {
      Test ob = new Test();
      int a, b;

      Scanner unos = new Scanner(System.in);
      System.out.print("Unesite a ");
      String ubroj = unos.next();
      a = Integer.parseInt(ubroj);
      System.out.print("Unesite b ");
      ubroj = unos.next();
      b = Integer.parseInt(ubroj);

    System.out.println("a i b prije poziva metode: " +
                     a + " " + b);

    ob.racunaj(a, b);

    System.out.println("a i b nakon poziva metode: " +
                     a + " " + b);
  }
}
```

HR 8:57 10.4.2014.


brojA
15
brojB
20
ob.izracunaj(int broj1, int broj2)
broj1
15
broj2
20
broj1 = *2
broj2 = /2
. . .
broj1
30
broj2
10

Prosljeđivanje argumenata
po referenci (call-by-reference)

- **za prosljeđivanje objekata**

Parametri koji predstavljaju objekte u osnovi sadrže samo referencu (put do) objekta.

Kada se ovo proslijedi metodi, parametar koji primi referencu ukazuje na isti objekt.

Primjer:

```
public static void utrostruči(Djelatnik x)
{ x.povećajPlaću(300); }
Poziv:
Djelatnik Mate = new Djelatnik(…):
utrostruči(Mate);
```

Prosljeđivanje argumenata
po referenci (call-by-reference)

Redoslijed događanja:

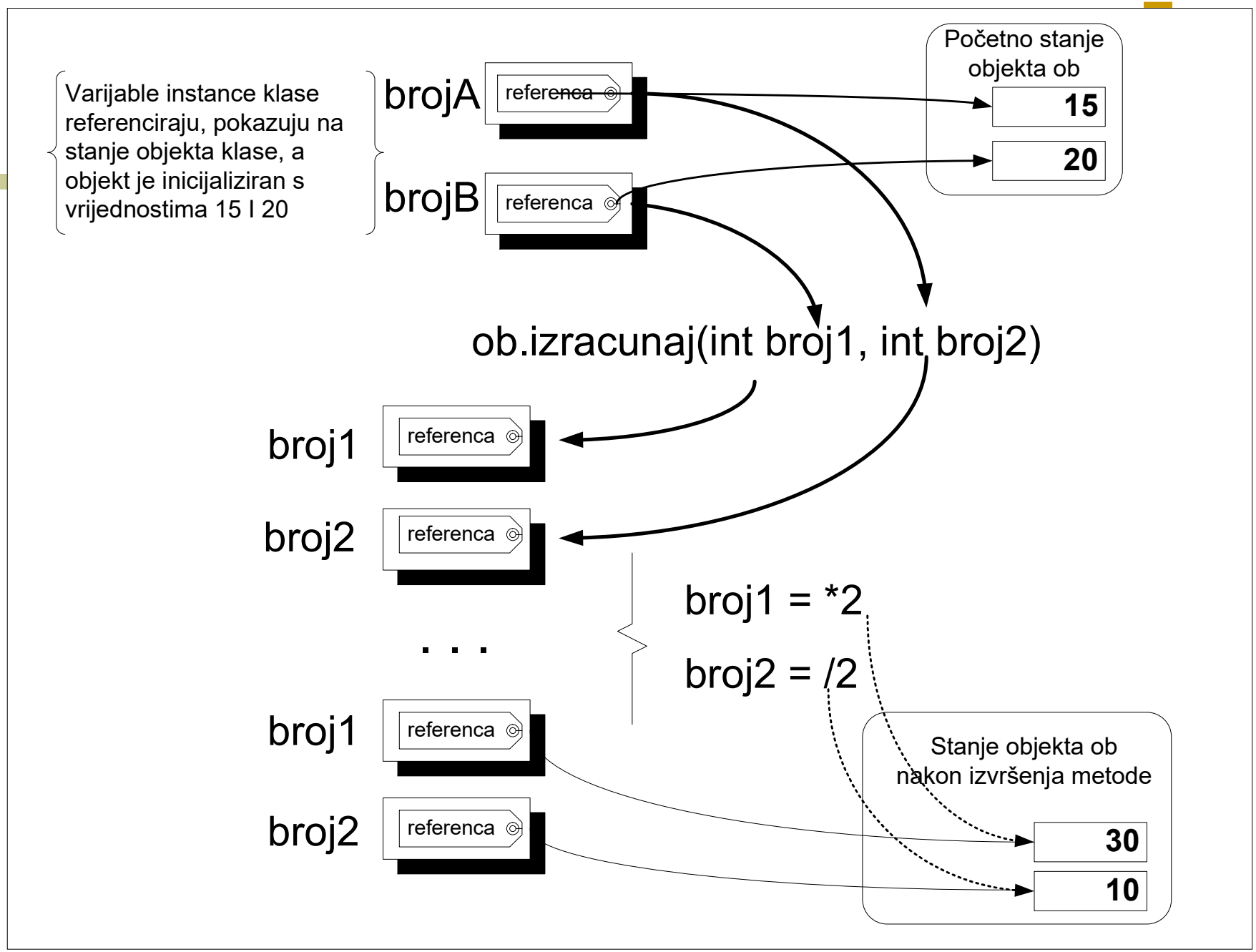
1. **“x” se inicijalizira kopijom vrijednosti “Mate”, tj. referencom na objekt**
2. **Metoda povećajPlaću se primjenjuje na tu referencu objekta. Objekt “Djelatnik” na kojega i “x” i “Mate” pokazuju (referenciraju) dobiva plaću uvećanu za 300%.**
3. **Metoda završava i parametarska varijabla “x” nije više u uporabi, ali objektna varijabla “Mate” i dalje pokazuje na objekt čija je plaća utrostručena.**

PozivPoReferenci.java – Blok za pisanje

Datoteka Uređivanje Formatiranje Prikaz Pomoć

```
class Test {
  int a, b;

  Test(int i, int j) {
    a = i;
    b = j;
     }

  // proslijedi objekt
  void racunaj(Test ob) {
    ob.a *=  2;
    ob.b /= 2;
    System.out.println("a i b unutar metode: " +
                       ob.a + " " + ob.b);
  }
}
class PozivPoReferenci {
  public static void main(String args[]) {

      int a, b;

      Scanner unos = new Scanner(System.in);
      System.out.print("Unesite a ");
      String ubroj = unos.next();
      a = Integer.parseInt(ubroj);
      System.out.print("Unesite b ");
      ubroj = unos.next();
      b = Integer.parseInt(ubroj);

      Test ob = new Test(a,b);

    System.out.println("ob.a i ob.b prije poziva: " +
                       ob.a + " " + ob.b);

    ob.racunaj(ob);

    System.out.println("ob.a i ob.b nakon poziva: " +
                       ob.a + " " + ob.b);
  }
}
```

HR 9:20 10.4.2014.


Varijable instance klase referenciraju, pokazuju na stanje objekta klase, a objekt je inicijaliziran s vrijednostima 15 I 20
brojA
referenca
brojB
referenca
Početno stanje objekta ob
15
20
ob.izracunaj(int broj1, int broj2)
broj1
referenca
broj2
referenca
. . .
broj1 = *2
broj2 = /2
broj1
referenca
broj2
referenca
Stanje objekta ob nakon izvršenja metode
30
10

Privatne metode

lako su metode većinom javne (public) mogu se definirati i kao privatne što znači da se mogu pozvati samo iz drugih metoda iste klase.

Kada se metoda proglasi privatnom ona ne mora biti raspoloživa ako dođe do promjene implementiranja.

BITNO: dokle god je metoda privatna programer može biti siguran da se neće nikada koristiti izvan operacije te klase.

Kada se odlučiti za privatnu metodu:

- **ako metoda nije bitna za korisnika klase**
- **Ako se metoda ne može jednostavno podržati ako dođe do promjene u implementiranju klase (specifični protokoli i sl.)**

Preopterećivanje konstruktora i metoda (constructor/method overloading)

Bitna osobina programskih jezika jeste uporaba imena.

Problem: preslikavanje nijansi svojstvenih ljudskim jezicima u programski jezik (homonim - jedna riječ više značenja).

Primjeri: ženska kosa, kosa crta, alatka kosa …

oprati košulju, oprati auto, oprati psa …

(Čekaj me dok dođem / čekaj me dok ne dođem)

Programski jezik: opratiKošulju košulju

opratiAuto auto

opratiPsa psa

Preopterećivanje konstruktora i metoda (constructor/method overloading)

Programski jezici zahtijevaju postojanje JEDINSTVENIH IDENTIFIKATORA za svaku funkciju.

JAVA

Konstruktori zahtijevaju mogućnost "preopterećenja" metoda.

RAZLOG: ime konstruktora je određeno imenom klase, što znači da može biti samo jedno. Ali, postoji potreba kreiranja objekta na više načina.

Preopterećenje konstruktora

```
public class Student7
{
 public static void main(String[] args)
 {
        student[] StudentR = new student[3];

        StudentR[0] = new student("Ana","Anić",1000);
        StudentR[1] = new student(1500);
        StudentR[2] = new student();


        for(int i=0; i<StudentR.length; i++)
        {
          student s = StudentR[i];
          System.out.println("Student "+s.uzmiime()+" "+s.uzmiprezime()+" "+"školarina
        "+s.uzmiskolarina());
       }
 }
}
```

```
class student
 { //prvi konstruktor
   public student(String i, String p, double s)
   { ime = i;
     prezime = p;
     skolarina = s;   }
   // drugi konstruktor
   public student(double s)
   {     this("konstruktor","2",s);    }
   // treći konstruktor - inicijalni
   public student()
    { /* inicijalne vrijednosti */ }
   public String uzmiime()
   { return ime; }
   public String uzmiprezime()
   { return prezime; }
   public String uzmitip()
   {return tip; }
   public double uzmiskolarina()
   {return skolarina; }
 private String ime, prezime, tip;
 private double skolarina;
   }
```

Preopterećenje konstruktora

```
class Mini extends Car {

    Color color;

    public Mini() {
        this(Color.Red);
    }

    public Mini(Color c) {
        super("Mini");
        color = c;
        // more initialization
    }

    public Mini(int size) {
        this(Color.Red);
        super(size);

    }
}
```

The no-arg constructor supplies a default Color and calls the overloaded Real Constructor (the one that calls super()).

This is The Real Constructor that does The Real Work of initializing the object (including the call to super())

Won't work!! Can't have super() and this() in the same constructor, because they each must be the first statement!

Ključna riječ: **this**

Ako postoji nekoliko konstruktora za klasu, postoji potreba za pozivom jednog konstruktora iz drugog kako bi se izbjegao dupli kod.

- ⇒**uporaba this operatora**
- ⇒**drugačije nego kod standardne uporabe**
- ⇒**znači eksplicitni poziv konstruktoru koji odgovara listi argumenata, što osigurava izravan način za poziv drugih konstruktora**
- ⇒**THIS mora biti prvi izraz u konstruktoru**

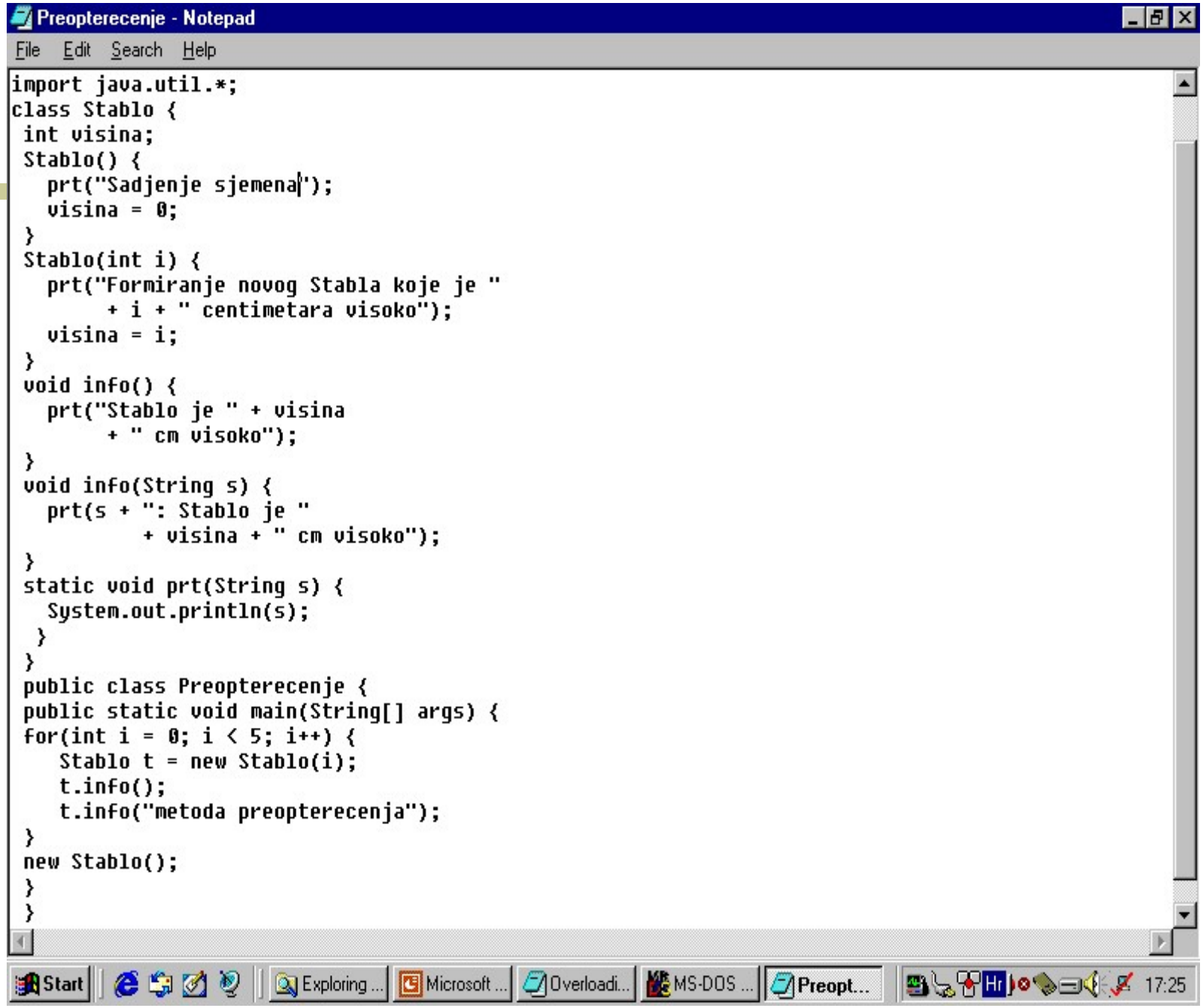
Preopterećivanje metoda (method overloading)

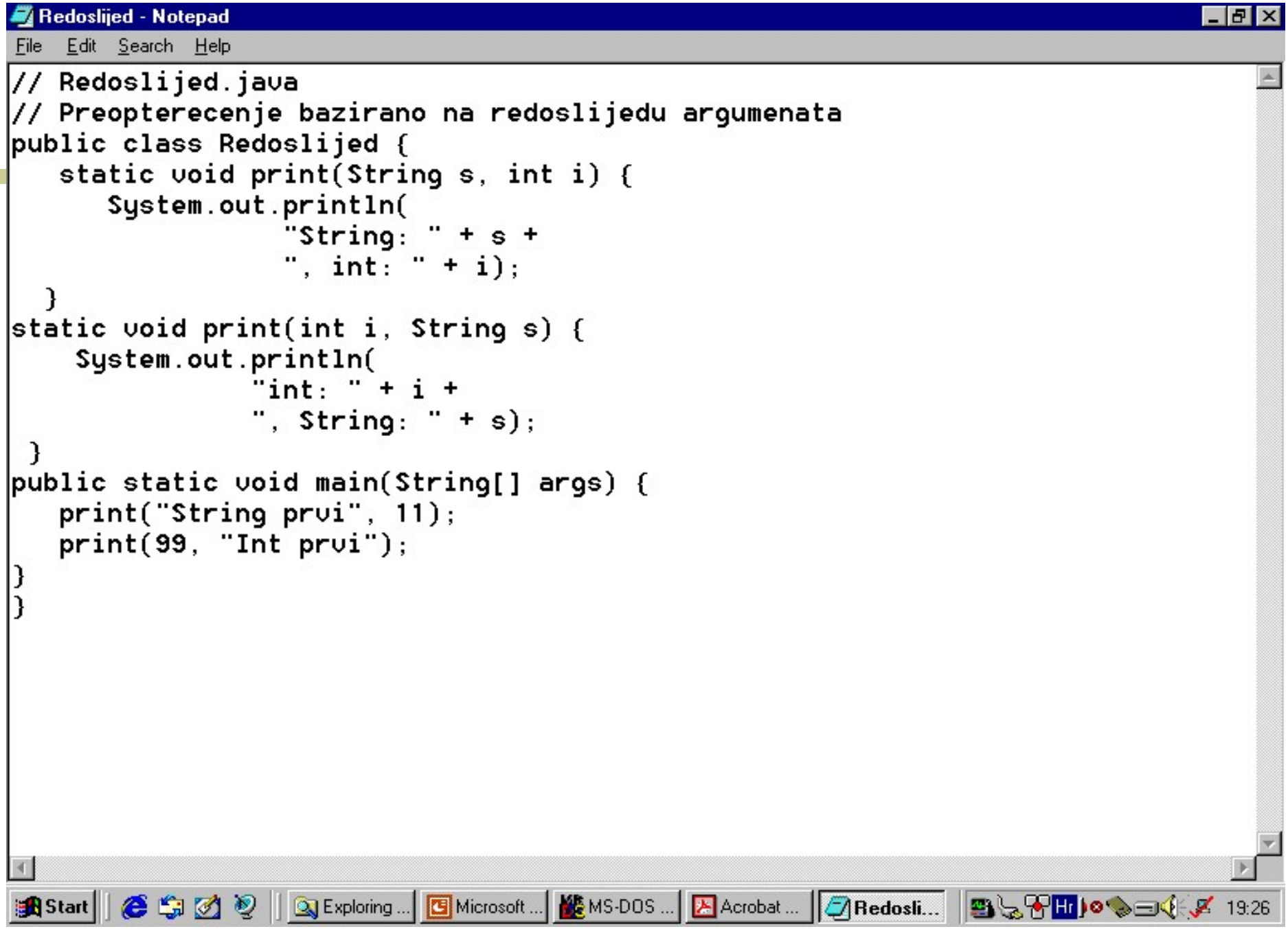
RAZLIKOVANJE

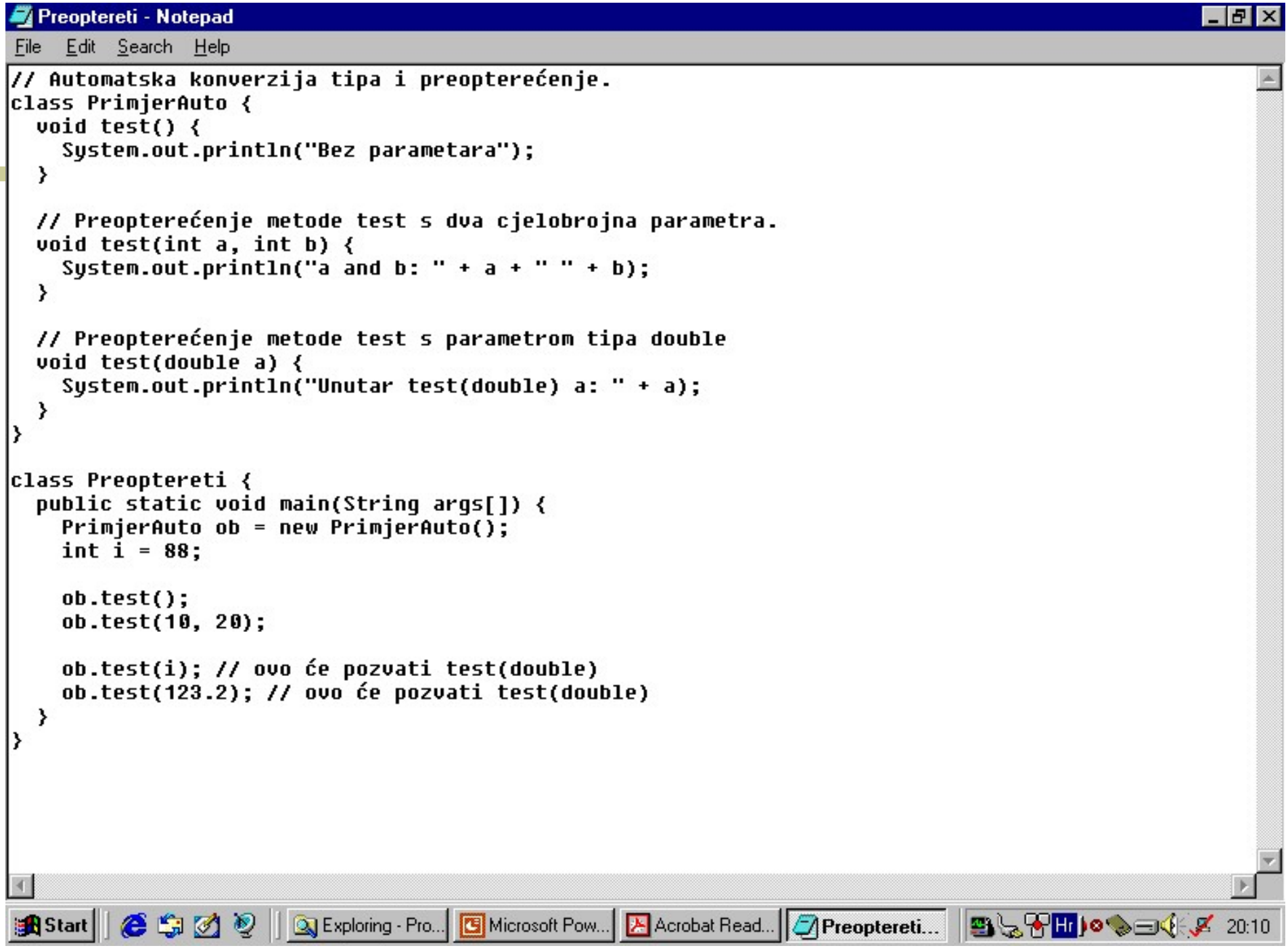
- **po tipu i/ili**
- **po broju argumenata**
- **po redoslijedu (ne preporučuje se)**

Povratni tip podataka nije dovoljan za razlikovanje metoda.

Slaganje ne mora biti potpuno točno – tu igra ulogu i automatska konverzija podataka.


Preopterecenje - Notepad
File Edit Search Help
import java.util.*;
class Stablo {
int visina;
Stablo() {
prt("Sadjenje sjemena");
visina = 0;
}
Stablo(int i) {
prt("Formiranje novog Stabla koje je "
+ i + " centimetara visoko");
visina = i;
}
void info() {
prt("Stablo je " + visina
+ " cm visoko");
}
void info(String s) {
prt(s + ": Stablo je "
+ visina + " cm visoko");
}
static void prt(String s) {
System.out.println(s);
}
}
public class Preopterecenje {
public static void main(String[] args) {
for(int i = 0; i < 5; i++) {
Stablo t = new Stablo(i);
t.info();
t.info("metoda preopterecenja");
}
new Stablo();
}
}
Start
Exploring ...
Microsoft ...
Overloadi...
MS-DOS ...
Preopt...
17:25


Redoslijed - Notepad
File Edit Search Help
// Redoslijed.java
// Preopterecenje bazirano na redoslijedu argumenata
public class Redoslijed {
static void print(String s, int i) {
System.out.println(
"String: " + s +
", int: " + i);
}
static void print(int i, String s) {
System.out.println(
"int: " + i +
", String: " + s);
}
public static void main(String[] args) {
print("String prvi", 11);
print(99, "Int prvi");
}
}
Start
Exploring ...
Microsoft ...
MS-DOS ...
Acrobat ...
Redosli...
19:26


Preoptereti - Notepad
File Edit Search Help
// Automatska konverzija tipa i preopterećenje.
class PrimjerAuto {
void test() {
System.out.println("Bez parametara");
}
// Preopterećenje metode test s dva cjelobrojna parametra.
void test(int a, int b) {
System.out.println("a and b: " + a + " " + b);
}
// Preopterećenje metode test s parametrom tipa double
void test(double a) {
System.out.println("Unutar test(double) a: " + a);
}
}
class Preoptereti {
public static void main(String args[]) {
PrimjerAuto ob = new PrimjerAuto();
int i = 88;
ob.test();
ob.test(10, 20);
ob.test(i); // ovo će pozvati test(double)
ob.test(123.2); // ovo će pozvati test(double)
}
}
Start
Exploring - Pro...
Microsoft Pow...
Acrobat Read...
Preoptereti...
20:10

PITANJA